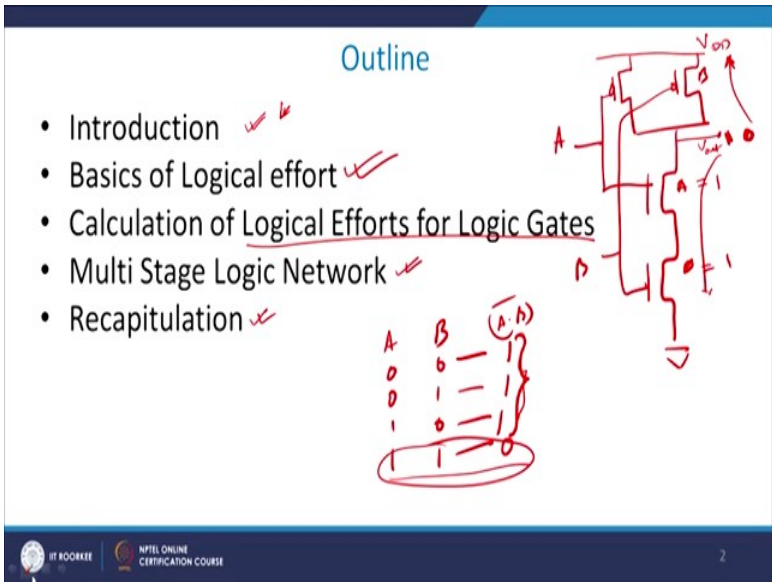
Microelectronics: Devices to Circuits
Professor Sudeb Dasgupta
Department of Electronics & Communication Engineering
Indian Institute of Technology Roorkee
Lecture – 22
Logical Efforts - I

Hello and welcome to the NPTEL online certification course on Microelectronics: Devices to Circuits. In our previous interactions we have seen that we have understood the basic concept of power dissipation on what issues these power dissipations can be lowered, how can they be lowered, we also looked into the concept of delay which means high to low and low to high propagation delay, we saw that the delay was obviously dependent on the width of the transistor beat pull-up or pull-down and therefore, higher the width lower was the value of a t_{PHL} or t_{PLH}, but the price you paid for it was that higher the width means more will be the capacitance and therefore, it will be difficult for the signal to drive the gate, right because your if your width increases the area under the gate increases and therefore, that is the problem area. So you need to optimize the design.

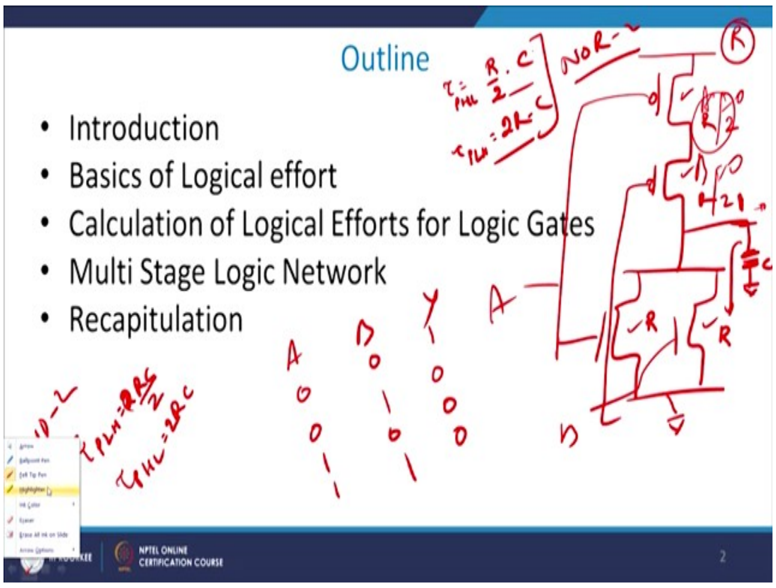
And in one of the optimization problems which we saw in our previous case we saw that, if you are able to keep a fan-out of approximately the stage ratio to approximately 3 we can get the minimum delay available to us. Keeping that in background let us start with one module which is a critical model for estimating delay between point A and point B of a circuit, right and that is what is the methodology which is adopted to do that is basically known as logical effort.

(Refer Slide Time: 2:12)



Outline

- Introduction
- Basics of Logical effort
- Calculation of Logical Efforts for Logic Gates
- Multi Stage Logic Network
- Recapitulation



Outline

- Introduction
- Basics of Logical effort
- Calculation of Logical Efforts for Logic Gates
- Multi Stage Logic Network
- Recapitulation

So today's topic will be referred to as a logical effort. So we will look at logical effort concept. So what will be doing is we will be introducing to you the logical effort what is the basic, sorry, I will give you the basic idea of logical effort and basic idea of logical effort, then we will be coming to, so we will introduce to you the logical effort what is the motivation for studying logical effort, and then what is the meaning of logical effort and where it is effectively used so basics of logical effort will be known to you.

Then how do you calculate logical effort for certain logic gates, right and then look at the multi stage network and then we will finally recapitulate, right. Before we move forward let me give you an idea about the concept or let me give you an idea about how you design it. For example if you have for example a gate which is something like this which is basically a two input NAND gate, right, right and you sometimes do like this and then you do like this then this is A and this is B, then if you put A and B as your inputs, right, then you will see for 0 0 since both A and B is, so this is A, this is B, so when both A and B are 0 output will be 1 when either one of them this is still output will be 1 and when both are 1 output will be 0. So this is your NAND gate approximation, so A dot B bar.

Now the idea here is that you can only have one transition from, so if you are storing 1 here you can only go to 0 provided both A and B are equals to 1, which is this one, but if this was initially 0, then you can go to 1 either of these 3, so if you fall back to your previous understanding you will know that as I discussed with you that 0 to 1 transitions are the transitions which are power consuming cycles. So therefore, if the… initially you have a data of 0 available with you, the probability that it will go to V_{DD} is much higher as compared to it going to down assuming that all the inputs are statistically possible and independent with respect to each other.

Then if initially you had 0 in your V_{out} output, then it can go to V_{DD} only if under all these three conditions, whereas it can stay to 0 only and 1 the condition which is basically this condition.

So you see that a power dissipation is therefore, also depends upon the statistics of your input and the type of gate whereas, if you look at this point and then if we let us say you design something like this that now your pull-up is basically your pull-up is basically series, right and you pull down is basically in parallel so if you have A here and you have B here so if you look here then if you plot A, B and Y, Y is the output here then if for this condition, right when both are 0 then only you will have 1, for all other cases you will have 0 which means that you initially if you have 1 here then the probability it to go to 0 is just that only one of the inputs or both of them should go to 1, so the probability is just to go to 0 is much higher whereas, if it wants to go to 0 then both A and B has to go to 0.

So again I am assuming that the, if it is statistically independent and equal probability of getting all the signals are available with you then I would not expect to see, I would expect to see power dissipating cycle more in a and NOR 2 logic as compared to a NAND 2 logic, this is the first observation. The second observation is that you see when you are pulling up you have two transistors in series, right and when you are pulling down in this case you have only one… even one transistor if switches ON you will be obviously in parallel.

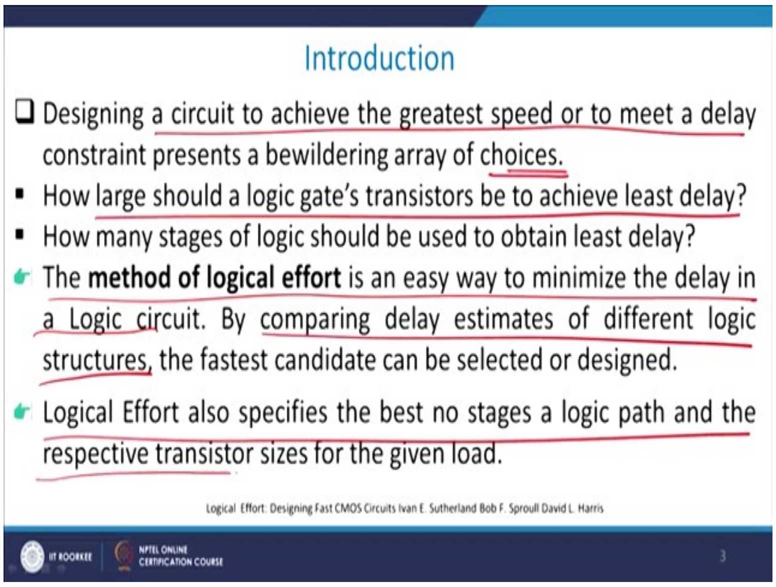
So, if we assume that each transistor is the resistance of R, then and if output is basically let us suppose is C, then I can safely say that if both the transistors are ON, both will be in parallel I will get τ equals to R by 2 right, into C whereas, if I assume this to be also R this to be also R, then τ this is $\tau_{PHL,}$ right, and this is τ_{PLH} will be equals to 2R times C. So means if I assume that NMOS's and PMOS's have exactly the same value of resistance as being offered during the ON state, then τ_{PHL} is RC by 2 whereas, τ_{PLH} is 2 RC, right and that is quite critical.

Therefore, low-to-high propagation delays 4 times larger as compared to high to low propagation delay. The same thing will reverse when you do a NAND-2 logic, this is for NOR-2. If you do a NAND-2 logic, for NAND-2 logic if you want to find out, then what will happen is that you will get an automatic reductions so τ_{PLH} will be equals to 2 RC and sorry

τ_{PLH} will be equals to RC by 2, and τ_{PHL} will be equals to RC twice RC I hope you understand the reason.

So therefore, if I for in this case if I want to make that my high to low and low to high are equal I need to size my PMOS in such a manner that by high to low and low to high are equal, so they will be in series so if I am able to make this one R by 2 and this one R by 2 then even in series these will add up, you will get R only and they will make me equal. So R by 2 primarily means that you make the width larger, but when you make the width larger then this capacitive loading also increases drastically, right. So this logical effort gives me an idea that what changes should I do in my circuitry in order to achieve the best possible results in terms of delay, right, with this basic introduction let me introduce to you the topic which we which we were supposed to do.

(Refer Slide Time: 8:44)



Introduction

- Designing a circuit to achieve the greatest speed or to meet a delay constraint presents a bewildering array of choices.
  - How large should a logic gate's transistors be to achieve least delay?
  - How many stages of logic should be used to obtain least delay?
- The **method of logical effort** is an easy way to minimize the delay in a Logic circuit. By comparing delay estimates of different logic structures, the fastest candidate can be selected or designed.
- Logical Effort also specifies the best no stages a logic path and the respective transistor sizes for the given load.

Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

IIT ROORKEE NPTEL ONLINE CERTIFICATION COURSE 3

So, as I discussed with you that designing a circuit that is what I was saying that designing a circuit to achieve the greatest speed or minimum delay will give you large number of choices, right? So you need to choose which is the best choice in front of you. Now the idea is therefore, how large should a logic gates transistor be to achieve the least delay? Please understand that just making it large does not make it… make that achieve the least delay because we just now saw that making it large also increase the capacitive loading and therefore, your capacitive loading will be larger.

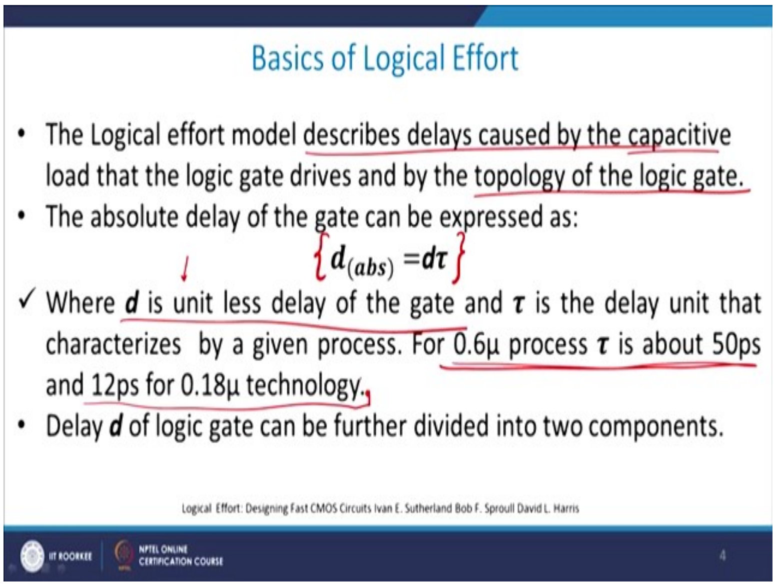
Now this is true for a single stage amplifier, single stage design, but generally in today's world you have multi stage circuits, right, so you have one stage driving another, driving another and so on and so forth. In such a scenario if you look very carefully the first stage is loaded by excessively by the subsequent stages, so the subsequent stage capacitive load will be so high that it will try to reduce the delay, try to enhance the delay of the first stage.

So can we therefore, find out the total number of stages which gives you the minimum delay? So what we say the method of logical effort if you look very carefully what is written here then the method of logical effort is an easy way to minimize the delay in a logic circuit, right, so what we do? We compare the delay estimates of different logic structures. For example we compare for a NAND-2, NOR-2, XOR-2 and then we say that the fastest candidate can be given by this particular design.

Logical effort also specifies the best number of stages in a logical path and the respective transistor sizes for a given load, so which means that if I know for sure that my output load is C_L or load is given to me which is exactly equals to some load then can we do a back calculation and say that okay my transistor size should be this much for this much load to be driven by an input to get the minimum delay, right?

So the overall justification or overall idea is that given a set of choice or given a set of paths or data path can we in some way or other find out a methodology by which we can optimize the delay, right? So this is what we get, right.

(Refer Slide Time: 11:22)



Basics of Logical Effort

- The Logical effort model describes delays caused by the capacitive load that the logic gate drives and by the topology of the logic gate.
- The absolute delay of the gate can be expressed as:

$$\left\{ d_{(abs)} = d\tau \right\}$$

✓ Where d is unit less delay of the gate and τ is the delay unit that characterizes by a given process. For 0.6µ process τ is about 50ps and 12ps for 0.18µ technology.
- Delay d of logic gate can be further divided into two components.

Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

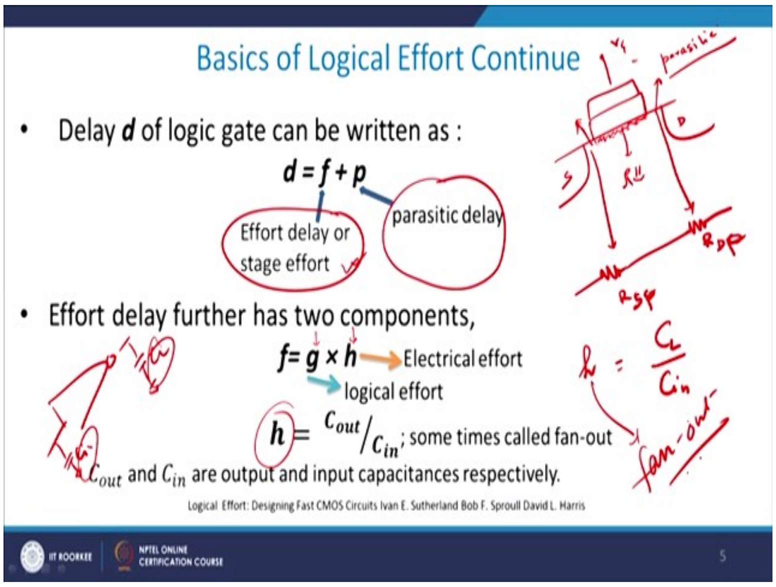
IIT ROORKEE | NPTEL ONLINE CERTIFICATION COURSE 4

Okay, so two things logical effect takes care of is what is the capacitive load right?, so we have a so you will have a delay because of capacitive load right, and you also have a delay by virtue of the topology of the logic gate, this we were discussing in the starting slide that there will be two delays associated with any design. The first delay will be the intrinsic delay which is basically because of the device, because of the capacitances and so on and so forth, right. Whereas, the second delay is basically depending upon the logic structure which you are using, are you using a NAND-2 logic, are you are using a NOR-2 logic, are you using what type of universal gate are you using in a digital logic in order to express it?

Now, the absolute delay of a gate can be expressed as, as you can see is given by this basic formula, where d absolute is equals to d multiplied by τ. Now d is basically unit less delay of the gate and it depends upon the type of process through which you are designing that particular gate, so this basically intrinsically you cannot do too much, you cannot do too much a manipulation in that because it is basically depending on the type of process the fabrication process through which the structure has gone through. So typically for 0.6 micron technology τ is approximately 50 microseconds, picoseconds and it is 12 picoseconds for 0.18 micron technology.

And that is quite interesting which means that as you go lower in the technology node, your intrinsic delay starts to lower down, so you see a lowering down of intrinsic delay of 50 picosecond to 12 picosecond, right. Now, so why do you multiply both of them? The reason being that, the reason you multiply is that τ is the delay unit, so if τ is the delay unit right, in picosecond, since τ is 1 picosecond and d is the delay unit by virtue of the fabrication, we can actually find out the total delay to be equals to d into τ, right.

(Refer Slide Time: 13:22)



Basics of Logical Effort Continue

- Delay ***d*** of logic gate can be written as :

$$d = f + p$$

Effort delay or stage effort

parasitic delay

- Effort delay further has two components,

$$f = g \times h$$

Electrical effort

logical effort

$h = C_{out}/C_{in}$; some times called fan-out

C_{out} and C_{in} are output and input capacitances respectively.

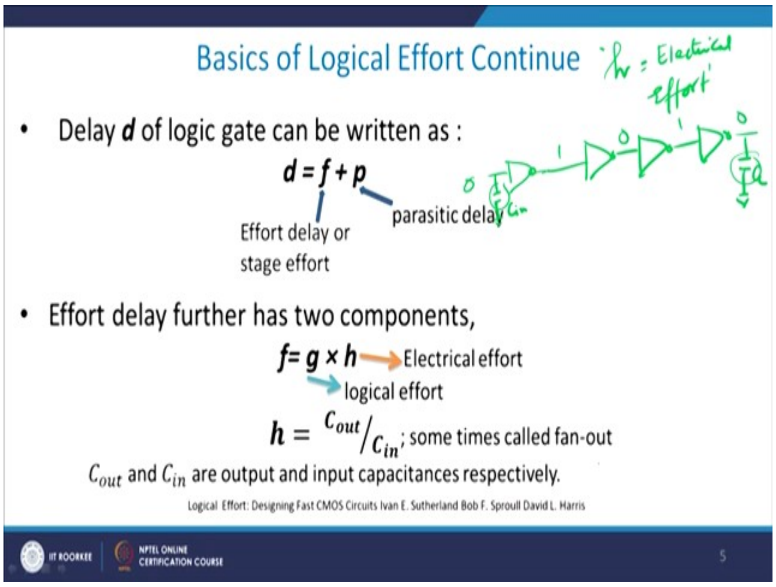
Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

Now, the delay of a logical gate d can be therefore, broken up into two parts, one is known as a stage effort or effort delay and another is the parasitic delay, what is a parasitic delay? Well parasitic delay is very simple straightforward we already discussed this point earlier also just to refresh your memories, I will just do it, let us suppose you have a gate here and you have a gate here and you have a source and drain here, right?

Now, what is happening is that the inversion layer is formed only below this region, so this is where the inversion layer is form and you have large carrier concentration here the resistivity here actually falls down, so resistance is lowered much here, but you see if you look at this region under also as under lap region these two regions, this, as well as this, then typically in these two regions you will automatically have a much much higher delay or the intrinsic delay will be larger because there are no charge carriers here and therefore, this will act as a parasitic, which means that it is basically adding so this sort will happen is something like this it will be one R here corresponds to this, and then you have a straight wire and then one R will be here, so this is R_1 and this is R_2 or R_S and R_D, so R_{SC}, R_{DC}, parasitic so R_{SP} and R_{DP}, so D_P is the drain side parasitic and S is the source side parasitic.

So the delay is by, one delay will be by virtue of this and you will also have a stage effort which is there this stage effort has got two component one is known as logical effort another is known as electrical effort, electrical effort is the most easiest one h and it is given as C_L or C_{out} by C_{in}. So, let us suppose I have a buffer or let me say a simple inverter and if my output load capacitance is C_L and my input load capacitance is C_{in}, then C_L this by this is effectively my h, right, this h is also sometimes referred to as fan out, right, so this is also referred to as sometime as fan out, C_{out} and C_{in}.

(Refer Slide Time: 15:42)



Basics of Logical Effort Continue

- Delay *d* of logic gate can be written as :

$$d = f + p$$

Effort delay or stage effort

parasitic delay

- Effort delay further has two components,

$$f = g \times h$$

Electrical effort

logical effort

$$h = C_{out}/C_{in}\text{; some times called fan-out}$$

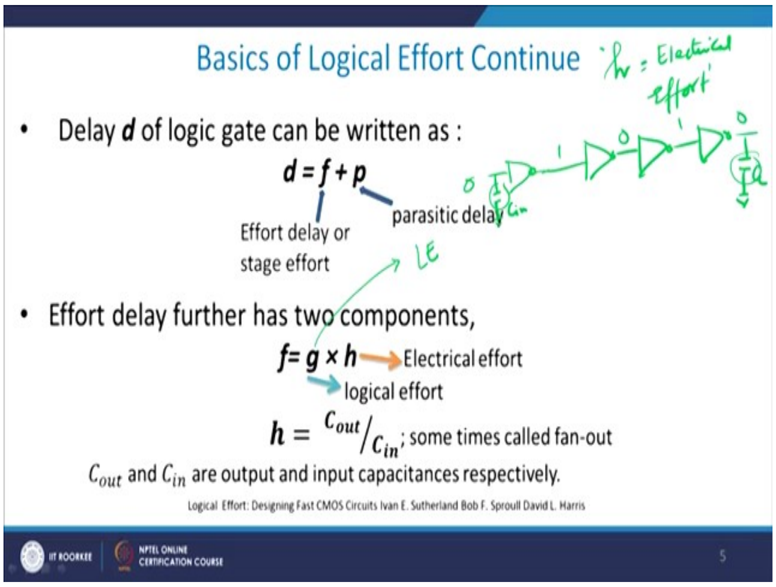
C_{out} and C_{in} are output and input capacitances respectively.

Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

Now, you can also have something like this that you do have let us suppose you have something like this that let me say, let me say you have so you have let us suppose a buffer here which means that you have let us say this, so I will get 1 0 1 0 and I will let us put a 0, I will get a 0, so this is a buffer. Now, in the buffer this is the load capacitance which you see C_L are external load and this is my C $_{in}$, so this, by this is referred to as the electrical effort, electrical effort French electrical effort is there.

So higher the electrical effort of course as you can see more will be the load capacitance as compared to C $_{in}$ and therefore, higher delay will be effectively large so f will be larger in that case if you get a higher electrical effort and that is the problem area which people face as far as electrical effort is concerned.

(Refer Slide Time: 16:46)



Basics of Logical Effort Continue

- Delay *d* of logic gate can be written as :

$$d = f + p$$

Effort delay or stage effort; parasitic delay

- Effort delay further has two components,

$$f = g \times h$$

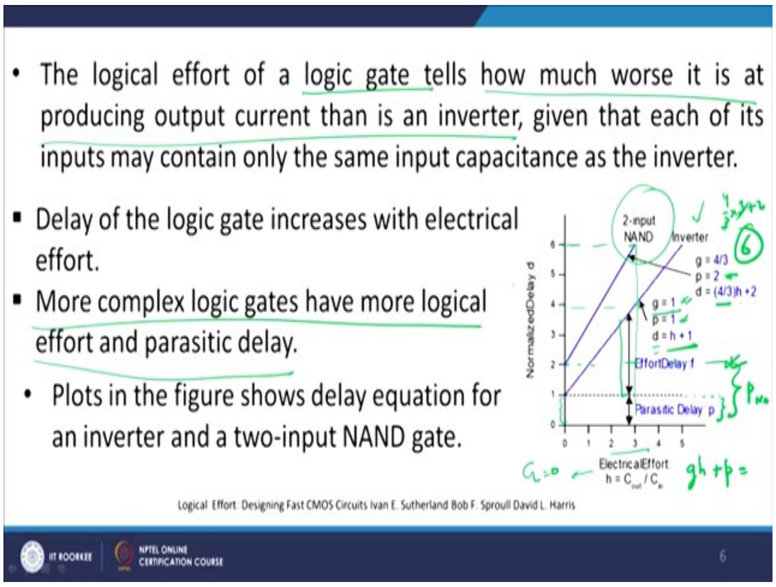
Electrical effort; logical effort

$$h = C_{out}/C_{in}$$; some times called fan-out

C_{out} and C_{in} are output and input capacitances respectively.

Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

h = Electrical effort
LE



- The logical effort of a logic gate tells how much worse it is at producing output current than is an inverter, given that each of its inputs may contain only the same input capacitance as the inverter.

- Delay of the logic gate increases with electrical effort.
- More complex logic gates have more logical effort and parasitic delay.
- Plots in the figure shows delay equation for an inverter and a two-input NAND gate.

Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

So what is a logical effort? Logical effort tells you that of a logical gate, logical effort for a logical gate tells you how much worse it is at producing output current than an inverter, right. So assuming that I have an inverter, I give an input to an inverter and I check out how much amount of current is flowing in the output side for a pair of inputs, then what I do I take the same pair of input and put it into a logical gate any logical NOR-2, NAND-2 whichever you want to calculate and see how much current is flowing, right.

If it is better good equal to an inverter this logical effort will let me know, right. As we have already discussed therefore, that logical gate, the delay in the logical gate will depend upon or will increase with the electrical effort, obviously it will do so, what is the electrical effort?

Because h, higher the value of h you will have higher will be the delay and therefore, why? Because your loading is typically very large, right.

And if you therefore, see a plot between electrical effort and normalize delay this graph here you see you will have a parasitic delay P is a constant one, this is independent of number of inputs, number of gates so on and so forth it is basically process and device centric and therefore, this parasitic delay is always constant. So even when your electrical effort is 0, you still have some amount of normalized delay. For example a two input NAND gate, we will talk about inverter later on. For example a two input NAND gate even with some electrical effort, electrical effort equals to 0, 0 means basically C_L equals to 0, I will automatically have this two, this two is because of primarily because of the because of the parasitic delay which you see.

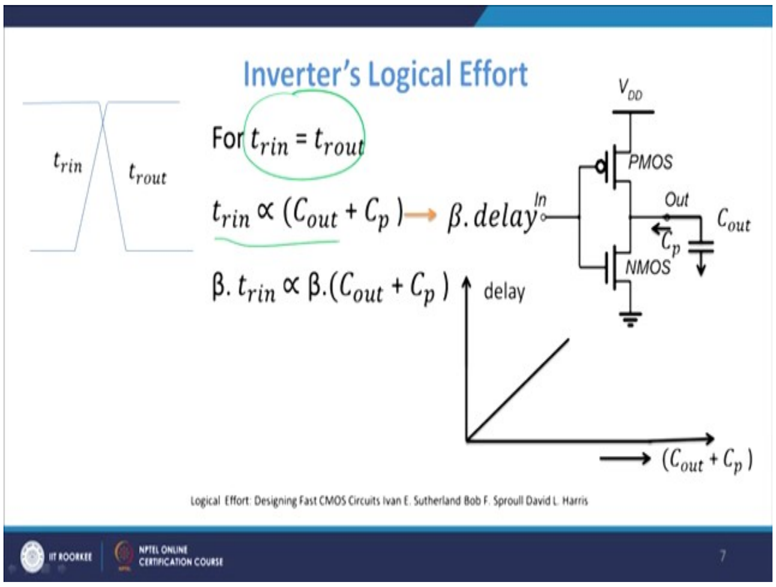
Now, if you go on increasing the electrical effort you automatically get a linear increase in the value of normalized delay and this is true also because then your load you in the external world is increasing and therefore, your delay will go on increasing, right. We define therefore, in this case, so let us suppose, let us suppose we take an inverter which is a skewed inverter, then we define this to be as a parasitic delay this p and this f is my effort delay, this f is my effort delay.

So at about approximately 3 the value of your normalized delay is 4 and we define g as equals to 1 and p equals to 1, what is g and what is g? If we look back to your previous slide, g is this, which is basically my logical effort, so g is my logical effort and if you go back to your here, what is p? The p value is the parasitic delay, parasitic delay is fixed always fixed, so I get f plus p equals to d, so what I am doing here is the p is 1, c_p is 1, what is g? Electrical effort is also let us assume it to be 1 because for our invert rate is 1, delay which you get capital D is equals to h plus 1, what is h? h is basically your electrical effort and you get d equals to h plus 1, right, that is how you work out.

And therefore, more complex logical gates have more logical effort, so if we have got rather than a two input NAND gate, if you have a three input NAND gate life becomes more difficult and therefore, your logical effort is larger in that case, right and you also have a larger parasitic delay if for example in that case. For example if you look back here a two input NAND gate you see your parasitics are actually 2, this is parasitic for a NAND gate, so you see p equals to 2, we will see how why is it g equals to 4 by 3, but p equals to 2, right.

The formula is g times h plus the parasitic delay is your total delay, so you see 4 into 3 into h plus 2, right, where h is the your electrical effort, right, so if you are plotting here somewhere here let us suppose you are plotting somewhere here, then here h equals to 3, 4 by 3 into 3 plus 2 is equal to 6, so normalized delay is approximately equals to 6, fine. How we got 4 by 3 for g, we will see that just in the next subsequent slide. So I get for 2 input NAND gate, now if it would have more complex and let me say it would have been 2 inputs XOR gate then the graph will change drastically in this case as compared to an inverter.

(Refer Slide Time: 21:18)



Inverter's Logical Effort

For $t_{rin} = t_{rout}$

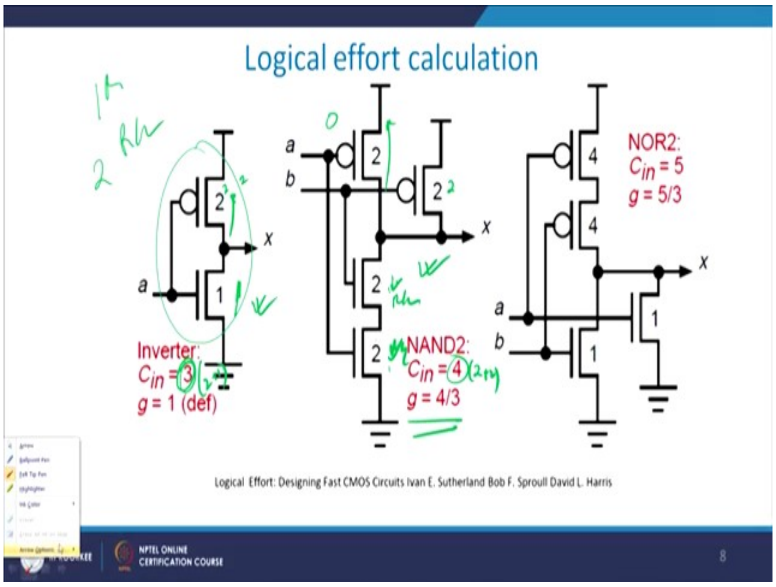
$t_{rin} \propto (C_{out} + C_p) \longrightarrow \beta.delay$

$\beta.t_{rin} \propto \beta.(C_{out} + C_p)$

Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

Let me come to the next slide and let us see how a logical effort works or in virtual logical effort assuming that my input rise time and output fall time exactly the same I can write down input rise time is proportional to C $_{out}$ which is this 1 plus C_P, C_P is the loading capacity which you see which is given as β times delay, β times delay or delay therefore, can be written as β times C $_{out}$ plus C_P and if you plot C $_{out}$ plus C_P versus delay you will always get a linear curve whose slope will be equals to beta. So this is a standard way of looking at it.

(Refer Slide Time: 21:46)



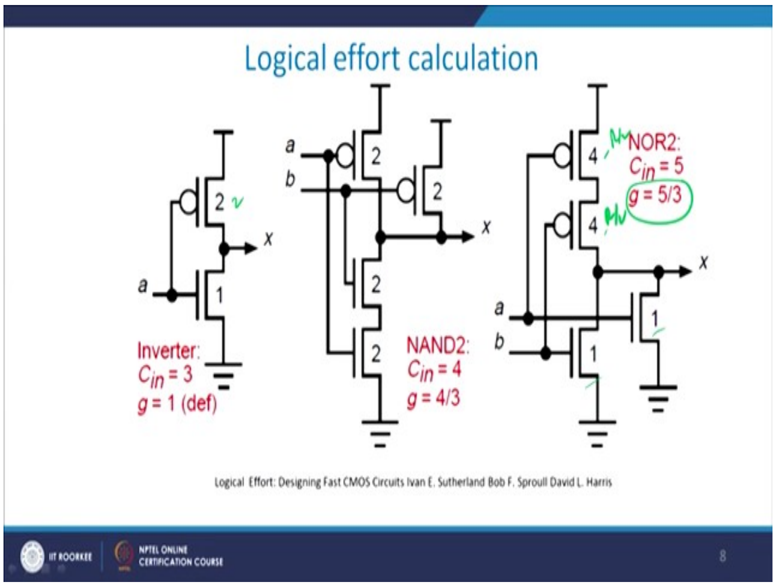
Logical effort calculation

Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

Now, how do you do a logical effort? You do it in this manner, that if you have a inverter which is basically as a single input inverter we make that PMOS is 2 times larger as compared to 1, so that t_{PHL} equal to t_{PLH}, now we try to make a NAND-2 logic and we have two PMOS's in parallel and 2 NMOS's is in series. Now, if you want this delay to be exactly equal to this delay, you have to make the lower transistor size double, then only I will get R by 2, R by 2 available to me, for pull up there is no issue because pull up this is 2, for any 0 I get 2 here and therefore, my pull up will be equal.

But for pull-down this is 1 and since this is 2, right, so this is 1, so if I have to double it because it is in series so this will be R, if 1 corresponds to R then 2 will corresponds to R by 2, so R by 2 plus R by 2 is always equals to 2 and the pull-down will be equal in that case. So pull up and pull down will be again 2 is to 1 available to you. So you see in this case C_{in} is equal to 3, why 3? Because 2 for PMOS 2 for PMOS and 1 plus NMOS, why is it 4 here? Because for single gate it is 2 for this thing and 2 for NMOS, right, so we define g to be 4 by 3, 4 by 3 means 4 is the effective capacitance seen from the NAND-2 logic, right divided by equally sized inverter which will give you the same delay, so 4 by 3, right.

(Refer Slide Time: 23:30)

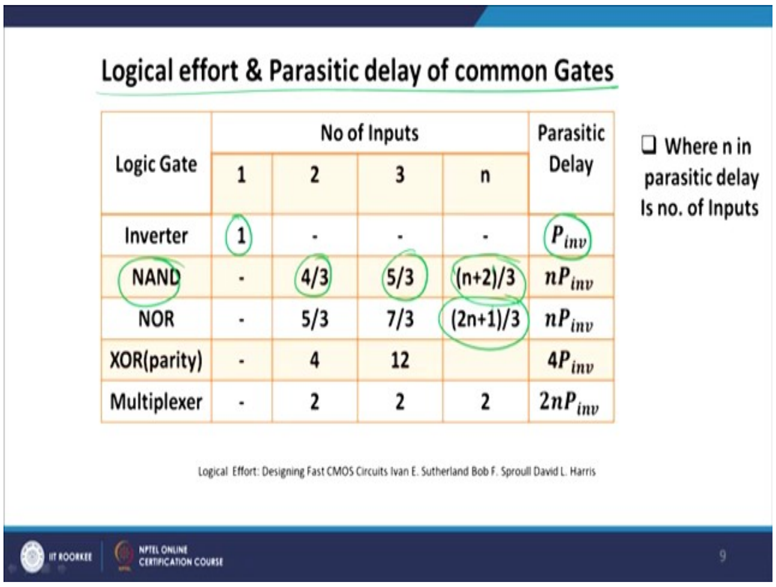


Logical effort calculation

Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

For a NOR-2 logic things change slightly and the reason is that they changed slightly, why? Because for NOR-2 logic understand if you want to compare this with this, then for pull-down I can afford to be 1 because both are in parallel so each one of them is 1 no problem, but in the pull-up I require this to be as 4 because then only it will become R by 2 because this is already 2, so this has to be R by, to make it R by 2, I have to double the size, so I am doubling the size so 4. So 4 means each block is looking at 4 plus 1, 5, so 5 by 3 is basically my g, which is which you see, right. So this is how you calculate the value of g, you compare that with a standalone inverter, right and that is how you get it.

(Refer Slide Time: 24:12)



Logical effort & Parasitic delay of common Gates

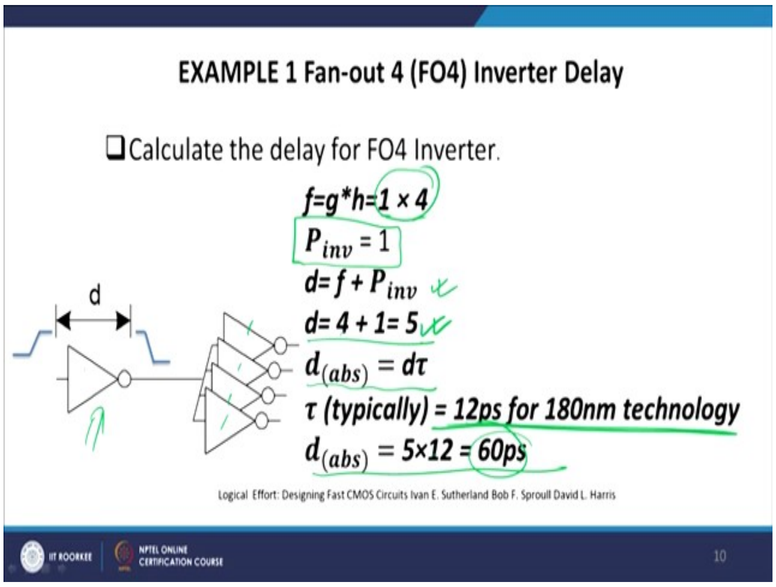
Logic Gate	No of Inputs				Parasitic Delay
	1	2	3	n	
Inverter	1	-	-	-	P_{inv}
NAND	-	4/3	5/3	(n+2)/3	nP_{inv}
NOR	-	5/3	7/3	(2n+1)/3	nP_{inv}
XOR(parity)	-	4	12		$4P_{inv}$
Multiplexer	-	2	2	2	$2nP_{inv}$

- Where n in parasitic delay Is no. of Inputs

Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

So if you look at the some of the logical efforts and common gates you see that inverter which is basically the 1 input obviously input has got only 1 for an inverter, parasitic delay is P inverter, for a NAND logic for 2 input 4 by 3, 3 input 5, so you see as the number of gates inputs increases the logical effort also increases, the general formula is n plus 2 by 3 for NAND gate and NOR gate it is 2n plus 1 by 3, where n is the number of inputs which you see right and that is a standard way of looking at it or a way of looking at it.

(Refer Slide Time: 24:44)



EXAMPLE 1 Fan-out 4 (FO4) Inverter Delay

- Calculate the delay for FO4 Inverter.

$f = g*h = 1 \times 4$

$\boxed{P_{inv} = 1}$

$d = f + P_{inv}$

$d = 4 + 1 = 5$

$d_{(abs)} = d\tau$

τ (typically) = 12ps for 180nm technology

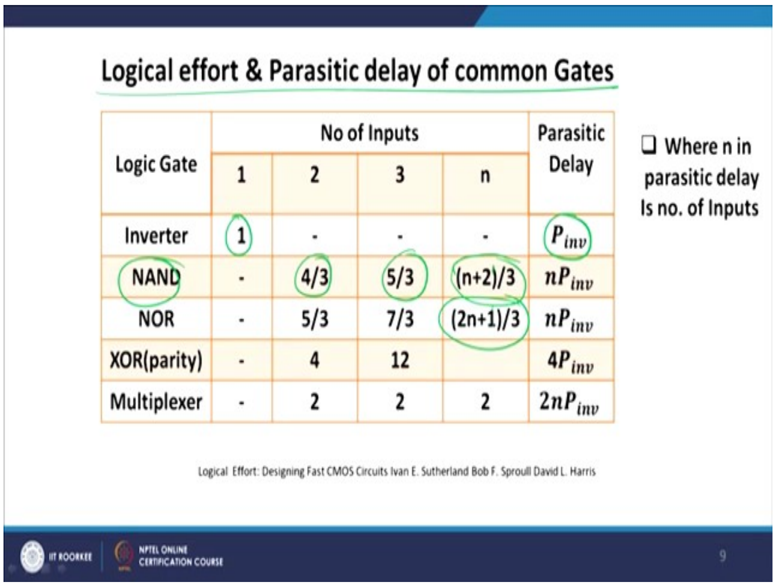
$d_{(abs)} = 5 \times 12 = 60ps$

Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

d
IIT ROORKEE
NPTEL ONLINE CERTIFICATION COURSE
10

I will give an example of FO4 inverter delay and if I have a inverter something like this then f is equal to g into h, g if you see single inverter it is 1, so 1 into assuming that you are driving 4 such inverters, so 1 into 4 is the output 1, P inverter is the parasitic delay assuming it to be equal to 1 I get d equals to f plus P inverter, right f is nothing but 4 so 4 plus 1 is 5, right so d absolute is equal to d into τ typically as I discussed with you 12 picoseconds for 180 nanometer so I get d absolute is equal to 5 into 12 which is 60 picoseconds so I got d from here multiply that with 12 I get 60, which means that if I have a single transistor driving 4 similar transistors with fan-out 4, I will get approximately 60 picosecond delay between the input and the output.

(Refer Slide Time: 25:34)



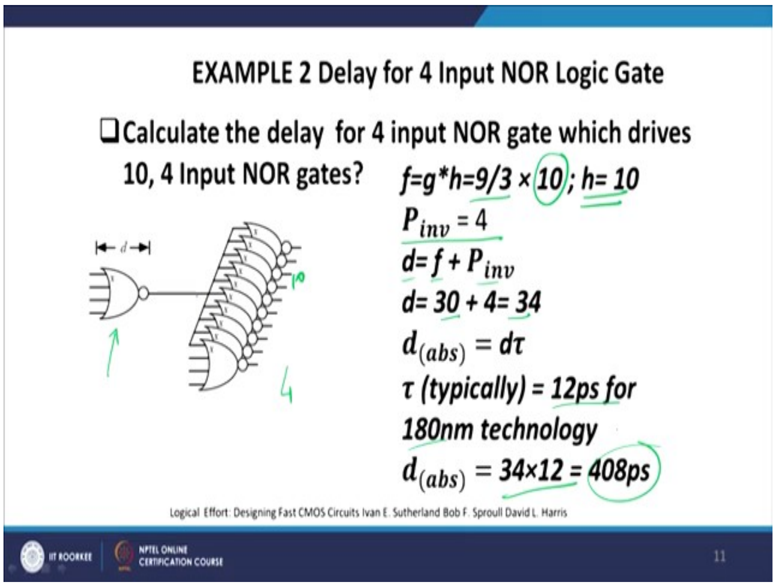
Logical effort & Parasitic delay of common Gates

Logic Gate	No of Inputs				Parasitic Delay
	1	2	3	n	
Inverter	1	-	-	-	P_{inv}
NAND	-	4/3	5/3	(n+2)/3	nP_{inv}
NOR	-	5/3	7/3	(2n+1)/3	nP_{inv}
XOR(parity)	-	4	12		$4P_{inv}$
Multiplexer	-	2	2	2	$2nP_{inv}$

❑ Where n in parasitic delay Is no. of Inputs

Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

9



EXAMPLE 2 Delay for 4 Input NOR Logic Gate

❑ Calculate the delay for 4 input NOR gate which drives 10, 4 Input NOR gates?

$f=g*h=9/3 \times 10$; $h= 10$

$P_{inv} = 4$

$d= f + P_{inv}$

$d= 30 + 4= 34$

$d_{(abs)} = d\tau$

τ (typically) = 12ps for 180nm technology

$d_{(abs)} = 34\times 12 = 408ps$

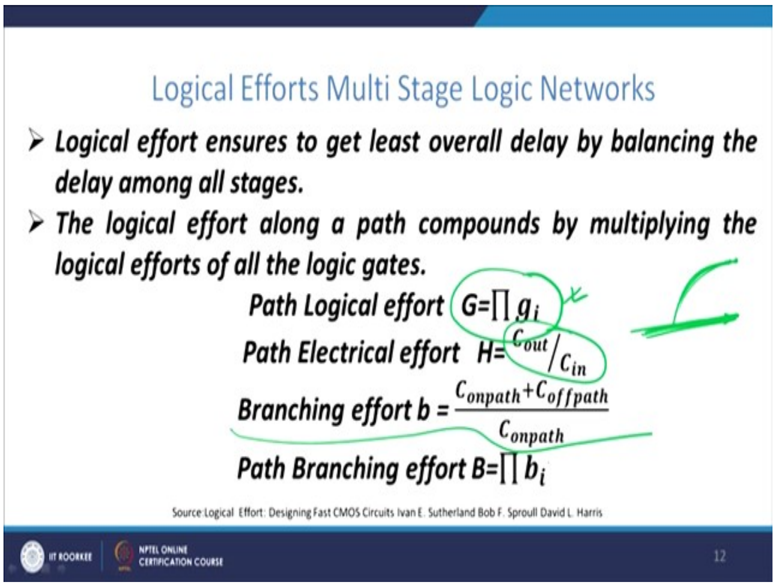
Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

11

Let us say an example of a NOR-2 logic, let us suppose I have a 4 input NOR logic driving 10 such gates, so as I discuss with you f will be g into h, g in this case will be 9 by 3, right, it will be 9 by 3 you can check it for a 4 input NOR gate for example just let me just show it to you for a 4 input NAND gate 4 2s are 8 plus 1, 9; 9 by 3 which you get, right, and that is what you get 9 by 3 multiplied by 10 because equally sized so h will be equals to 10, right so I get and let us suppose the parasitic for inverter is 4, so I get d equals to f plus inverter so I get 30 plus 4; 34 and for again 180 nanometer 34 into 12 is 408 picosecond.

So therefore, a 4 input NOR driving 10 such 4 input NOR gate the picosecond 408 picosecond is the gate, so first you have to find out the f value, then you find out the if P parasitic is given to you, d equals to f plus p inverter when you once you find out the value of d you then find out multiply it by τ, τ is always fixed for a particular technology and from there you can calculate the total delay between the gates.

(Refer Slide Time: 26:55)



Logical Efforts Multi Stage Logic Networks

- ***Logical effort ensures to get least overall delay by balancing the delay among all stages.***
- ***The logical effort along a path compounds by multiplying the logical efforts of all the logic gates.***

Path Logical effort $G=\prod g_i$

Path Electrical effort $H=C_{out}/C_{in}$

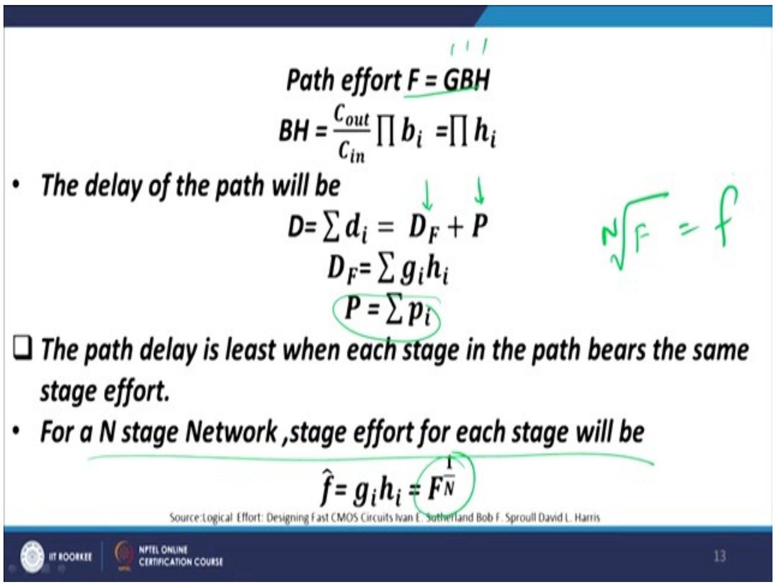
Branching effort $b = \frac{C_{onpath}+C_{offpath}}{C_{onpath}}$

Path Branching effort $B=\prod b_i$

Source:Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

So what we will, what we will try to do is give you a brief idea about the logical effort for a multi, so path stage a path so now if you have a large path then we what we do is that we define a path logical effort, so in a large path we go on adding the logical effort for each one of them, we define path electrical effort as C $_{out}$ by C $_{in}$, we have already discussed this path, we define a new term which is known as branching effort which is C $_{on}$ path versus plus C $_{off}$ path by C $_{on}$ path, which means that let us suppose I have this and my signal goes via this path, then this is C $_{off}$ path and this is C $_{on}$ path, so we see the C $_{on}$ path plus C $_{off}$ path thereby C $_{on}$ path gives you this value and the total branching effort is given by this formula.

(Refer Slide Time: 27:34)



Path effort $F = GBH$

$$BH = \frac{C_{out}}{C_{in}} \prod b_i = \prod h_i$$

- *The delay of the path will be*

$$D = \sum d_i = D_F + P$$

$$D_F = \sum g_i h_i$$

$$P = \sum p_i$$

❑ *The path delay is least when each stage in the path bears the same stage effort.*

- *For a N stage Network, stage effort for each stage will be*

$$\hat{f} = g_i h_i = F^{\frac{1}{N}}$$

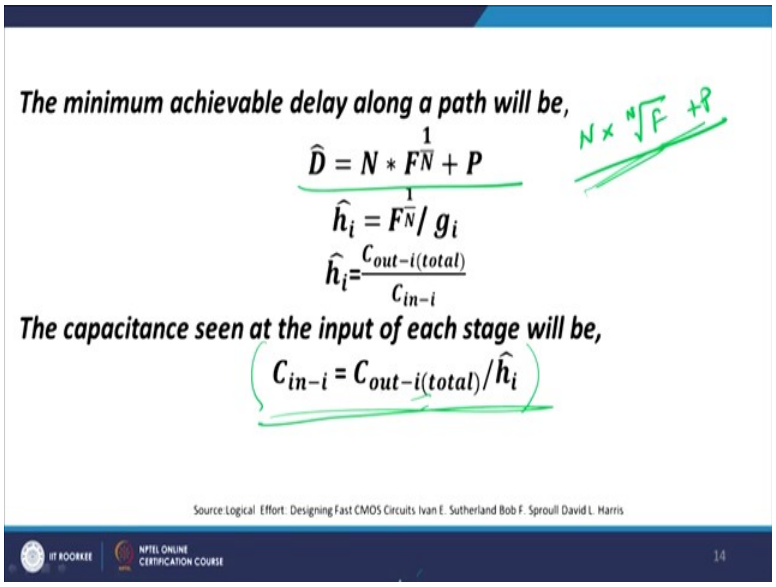
$$\sqrt[N]{F} = f$$

Source:Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

13

From these explanations I get that total logical path will be P_i, P_i will be the individual stage so if you add all the individual stages you get the total branch effort, so what we do is, we different path effort to be equal to g b into h, right and from there we find the delay as this plus p which is p is the parasitic delay for each stage, right? We therefore, say for N stage network, stage effort for each stage will be F to the power 1 by n so the square root or nth root of F is my stage effort which we see, right?

(Refer Slide Time: 28:10)



The minimum achievable delay along a path will be,

$$\hat{D} = N * F^{\frac{1}{N}} + P$$

$$N \times \sqrt[N]{F} + P$$

$$\hat{h}_i = F^{\frac{1}{N}} / g_i$$

$$\hat{h}_i = \frac{C_{out-i(total)}}{C_{in-i}}$$

The capacitance seen at the input of each stage will be,

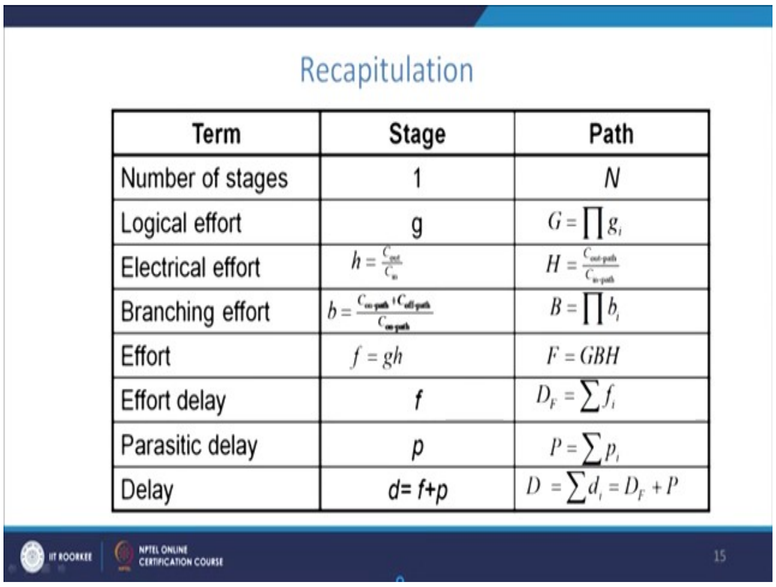
$$C_{in-i} = C_{out-i(total)} / \hat{h}_i$$

Source:Logical Effort: Designing Fast CMOS Circuits Ivan E. Sutherland Bob F. Sproull David L. Harris

14

So what we do is we delay we define as N into F, so N into nth root of F plus P is basically by delay, right and that is what you get the total delay minimum achievable delay along a path, right. Now, so the minimum capacity C_{in} where capacitance C_{in} at each stage will be given by this formula, where C in is basically the input capacitance of ith stage and I get total in the output stage for the ith stage I get C out divided by h_i cap, h_i cap is the basically the logical effort this F root N divided by logical effort.

(Refer Slide Time: 28:45)



Recapitulation

Term	Stage	Path
Number of stages	1	N
Logical effort	g	$G = \prod g_i$
Electrical effort	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out\text{-}path}}{C_{in\text{-}path}}$
Branching effort	$b = \frac{C_{on\text{-}path} + C_{off\text{-}path}}{C_{on\text{-}path}}$	$B = \prod b_i$
Effort	$f = gh$	$F = GBH$
Effort delay	f	$D_F = \sum f_i$
Parasitic delay	p	$P = \sum p_i$
Delay	$d = f + p$	$D = \sum d_i = D_F + P$

So, let me recapitulate therefore, if we have logical effort g, electrical effort as h, branching effort as b, then we define the total effort f is equal to g into h and effort delay to be equals to f and p and therefore, the total delay is always equals to f plus p, how do you find out f? g into h you find out, so in a single stage, so in multiple stage you can do it, so first of all find this and then you multiply with this and then you then you add to p and you get the total delay, right.

So we have understood two important points, I will come into details of this one at later course also, but at this stage we actually understood that given that means if a particular logic gate is driving some other logic gate, right maybe it is a N input logic gate we can find out the delay between point A and point B for the design, right that will help you to gain in this area, with this let me thank you for your hearing, thanks a lot, thank you!